PASSWORD MANAGEMENT

BACKGROUND OF THE INVENTION

The present invention relates generally to data processing systems, and more specifically to password management.

In a centralized or distributed data processing system, a plurality of systems or services (collectively "resources") may be available to a user. Each of these resources may have an access control, requiring a user to have a valid user identification ("user ID"), as well as an authenticator, such as a valid key, token or password, to gain access. For the purposes of the present description, the term "password" is used in its broadest sense to cover any such authenticator. For a user requiring access to a number of resources, remembering and entering a user ID and password for each resource at the beginning of each logon session may be cumbersome. The problem may be exacerbated if there are multiple password management systems being used to manage each password. A solution for addressing this problem would be desirable.

SUMMARY OF THE INVENTION

The present invention provides a password management solution which provides a user with convenient access to multiple resources (e.g. systems and services), and also provides the flexibility to establish varying password security requirements for each resource.

In an embodiment, there is provided a password registry for registering resources and securely storing encrypted passwords and associated identifying information. The identifying information may include, for example, a user identification (user ID), a resource hostname, and a resource type.

An unencrypted user-provided password may be encrypted by a process associated with each resource, using an encryption algorithm specific to that resource, before storage of the encrypted password in the password registry. An encrypted password retrieved from the password registry may be decrypted by a process associated with each resource using a decryption algorithm specific to that resource.

In an embodiment, in a distributed computing system, an encryption/decryption process may execute as a "front-end" client process running locally with the password registry, and may control access to a "back-end" resource.

In an embodiment, the "front-end" client process and the password registry may run on a local "workstation" which may be used to connect to a remote "back-end" resource server. For the purposes of the present description, the term "workstation" is used in its broadest sense to describe any local system on which the "front-end" client process may run.

A user interface may be provided to manage the passwords and associated identifying information stored in the password registry.

In an aspect of the invention, there is provided a method of managing a user's passwords for a plurality of resources using a password registry associated with said user, comprising:

(i) encrypting an unencrypted user-specified password at a process associated with said each resource;

(ii) receiving an encrypted password from said process associated with said each resource;

(iii) storing said encrypted password in said password registry, such that said unencrypted user-specified password is unknown to said password registry.

In another aspect of the invention, there is provided a method of managing a user's passwords for a plurality of password protected resources accessed from a workstation over a network, comprising:

at a workstation process associated with a network accessed password protected resource:

receiving a user selected password;

encrypting said user selected password as an encrypted password;

storing said encrypted password in a password registry.

In yet another aspect of the invention, there is provided a computer readable medium having computer readable program code embedded in the medium for managing a user's passwords for a plurality of resources accessed from a workstation over a network, the computer readable program code including:

code for establishing a process at a workstation, said process acting as a front-end for a network accessed resource;

code for enabling said process to receive a user-specified password;

code for enabling said process to encrypt said user-specified password as an encrypted password and output said encrypted password, in association with identifying information, to a password registry;

code for enabling said process to receive a request from a workstation user to access said resource and to, in response, obtain said encrypted password from said password registry using said identifying information.

In another aspect of the invention, there is provided a system for managing a user's passwords for a plurality of password protected resources accessed from a workstation over a network, comprising:

at a workstation process associated with a network accessed password protected resource:

means for receiving a user selected password;

means for encrypting said user selected password as an encrypted password;

means for storing said encrypted password in a password registry.

These and other aspects of the invention will be apparent from the following more particular descriptions of exemplary embodiments of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

In the figures which illustrate exemplary embodiments of this invention:

FIG. 1 is a schematic block diagram of an illustrative operating environment for exemplary embodiments of the invention.

FIG. 2 is a schematic block diagram of an exemplary embodiment.

FIG. 3A is a further schematic block diagram of an exemplary embodiment.

FIG. 3B is a further schematic block diagram of an exemplary embodiment.

FIG. 3C is a further schematic block diagram of an exemplary embodiment.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

Referring to FIG 1, shown is an illustrative distributed data processing system 100 which may provide an operating environment for exemplary embodiments of the invention. A plurality of resources (e.g. systems 110a – 110d and services 112a – 112d) may be connected via suitable connections 114a – 114d to a network 120. A user workstation 130 may also be connected to the network 120 via a suitable connection 122. The user workstation 130 may include a network I/O module 124 for receiving the connection 122. The user workstation 130 may allocate data processing resources to a user workspace 200. The user workspace 200 may be embodied, for example, as a process running on a central processing unit ("CPU") in the user workstation 130. As shown in FIG. 1, the user workspace 200 may access a storage disk 160 via a storage I/O 162, and a memory 170. The user workspace 200 may be accessed by a user from a user interface 150 connected via a user interface I/O module 152.

In an embodiment, the user workspace 200 may include a plurality of processes 212a – 212d which may be associated with the resources (systems 110a – 110d or services 112a – 112d). In the illustrative operating environment of FIG. 1, the processes 212a – 212d may be considered as "front-end" clients to various "back-end" resource servers.

The user workspace 200 may further include a password registry 210. In an embodiment, the password registry 210 may be embodied as a process running in the user workspace 200 and have a corresponding file for storing information on the storage disk 160.

The user workspace 200 may also include a user interface process 154 for facilitating access via the user interface 150. As will be explained, the user interface process 154 may provide access to the password registry 210 for various password management functions.

Given the illustrative operating environment of FIG. 1, an exemplary embodiment in use is now described.

In the exemplary embodiment, the user workstation 130 may be used, for example, to run an integrated application development environment, or more simply, an "IDE". In the present example, the user workspace 200 may be the IDE running on the user workstation 130. The front-end processes 212a – 212d may then provide an interface for various application development services 112a—112d which may be "plugged" in as extensions to the IDE.

For example, a commercially available IDE product known as the Eclipse™ workbench allows various application development tools from a number of vendors to be integrated into a single IDE. A specific example of a development tool which may be integrated into the Eclipse workbench is the Remote System Explorer ("RSE") in the commercially available "WebSphere™ Development Studio Client ("WDSc") for iSeries", which allows users to browse a file system, run commands, and view jobs on a remote iSeries / Linux / Unix or Windows system.

In an embodiment, a registration mechanism may be used by each tool vendor to register their tools with the password registry 210. In the illustrative example shown in FIG. 2, there are a number of different types of development tools which are registered with the password registry 210: tool 1, which is a tool for accessing a remote "iSeries" system type; tool 2, which is a "Linux" type; tool 3, which is a "Database" type; and tool "x" which is a "SCM" (Source Configuration Management) type. In an embodiment, each of these tools may have corresponding "front-end" process 212a—212d, respectively, running in the user workspace 200.

In the illustrative example, an Eclipse extension point may be provided so that each tool vendor can access the password registry 210. For further information on Eclipse extension points, the reader is directed to the Internet URL "eclipse.org". When implementing such an extension point, two pieces of information may be required: The first piece of information is the specific resource "type" for which a tool would like to store password information. The second piece of information is a "module" which handles encrypting passwords for each resource.

In an embodiment, a number of application programming interfaces ("APIs") may be provided:

a) An API for querying the password registry 210 for the encrypted password for a given user ID and resource.

b) An API for storing a new user ID / password pair in the password registry 210 and removing or changing an existing pair.

c) An API for enabling the password registry 210 to ask a registered tool (e.g. a front-end process 212a—212d) to encrypt a new password entered by the user.

In an embodiment, the first two API's, namely a) and b), may be provided by the password registry 210. The third API may be implemented by the "module" which handles encrypting passwords for each resource.

The password registry 210 never stores an unencrypted password for any of the tools. Instead, before a password is stored, the password is encrypted by each corresponding front-end process 212a—212d running in the user workspace 200.

Thus, each tool vendor can establish its own password security requirements, using whatever password encryption/decryption algorithm it wants or requires. The password registry 210 may then store the encrypted passwords regardless of the encryption algorithm used by each tool vendor. When an encrypted password is retrieved from storage 160, the encrypted password may be decrypted by a corresponding front-end process 212a—212d.

An illustrative example involving one of the development tools of FIG. 2 is now described.

Referring to FIG. 3A, in an embodiment, when a user uses the user interface process 154 to add a new password, the user may be asked to provide the following pieces of identifying information: a) user ID; b) resource hostname; c) resource type; and d) an unencrypted password. A suitable interface to enter this identifying information may be provided by the user interface process 154. For example, when indicating the resource type, the user may select this from a drop down menu provided by the user interface showing all registered tools (e.g. iSeries, Linux,

Database, SCM). Once the user has entered this information, the password registry 210 initiates communication with one of the corresponding front-ends 212a-212d.

In this example, the password registry 210 delegates to the selected front-end 212b the task of encrypting the unencrypted user-specified password, using the encryption module previously identified during registration. In an embodiment, a version number may be provided with the password, so that if the encryption algorithm is changed in a future release of the tool 212b, old passwords may be migrated to the new encryption / decryption algorithms.

Still referring to FIG. 3A, after the unencrypted user-specified password is encrypted by the front-end 212b, the password registry 210 may write the user ID, resource hostname, resource type, encrypted password, and optionally the encryption/decryption version number, to the storage disk 160.

Referring to FIG. 3B, when a user requires access to a resource, the user may initiate access directly with a front-end process 212a—212d. In the example shown in FIG. 2, the user may initiate access to the Linux tool via the corresponding front-end process 212b. This user-initiated access attempt may prompt the front-end process 212b to query the password registry 210 to see if an encrypted password for that front-end process 212b is available in the password registry 210. A "query key" used by the front-end process 212b for this purpose may consist, for example, of the user ID, resource type, and resource hostname. The password registry 210 in turn may access the disk 160 to determine if the corresponding encrypted password is stored on the disk 160. If a stored, encrypted password exists for this query key, then the encrypted password may be retrieved by the password registry 210 from the disk 160. The password

registry 210 may then pass the encrypted password back to the front-end process 212b for decryption. A similar, corresponding access method may be used to access each of the other resources, in turn, via their respective front-ends 212a, 212c, and 212d.

In each case, since it is the front-end process 212a – 212d that originally encrypted the password (FIG. 3A), the front-end process 212a – 212d may also be used to decrypt the retrieved, encrypted password and allow access to an authorized user.

Only an authorized user should be able to access a password registry 210 associated with that user. For example, the user workstation 130 and/or the user workspace 200 may have its own operating system-based secure access, such that the password registry 210 containing the encrypted passwords is only available upon authorized access to the workstation 130 and/or the workspace 200. As the passwords are stored in an encrypted form that can only be decrypted by the original encrypting front-end process 212a—212d, any unauthorized access to the encrypted passwords stored on the disk 160 should not pose a risk.

In an embodiment, in order to prevent an encrypted password from being used by an unauthorized user from another workstation (not shown), the encryption key used for encrypting the password may include some form of workstation specific information so that the password registry 210 cannot be used on a different workstation. For example, a unique TCPIP address of the workstation 130 may be utilized in the encryption key.

Referring to FIG. 3C, when a user initiates access with a front-end (e.g. front-end 212b as in FIG. 3B), but an encrypted password is not found for the query key, then the password registry

210 may notify the front-end 212b. The front-end 212b may in turn notify the "back-end" (e.g. one of the services 112a—112d systems 110a—110d), which may in turn prompt the user to enter a user ID and password via the password registry 210. In this case, the tool vendor can use an API provided by the password registry 210 for storing this information onto the disk 160. Prior to such storage, each front-end process 212a – 212d may encrypt the unencrypted user-specified password using a specific encryption algorithm, as shown in FIG. 3A.

In an embodiment, the user can also access the password registry 210 via the user interface process 154 to store, modify, and delete information for accessing each back-end resource (e.g. systems 110a—110d and services 112a—112d). In each case, the password registry 210 will not store an unencrypted password on the disk 160, and it will be the front-end process 212a – 212d that encrypts and decrypts the passwords based on a specific encryption/decryption algorithm.

While a distributed data processing system has been described in the above example, the invention may be practiced in a centralized data processing system in which multiple passwords and user IDs are required for secure storage in a password registry. In this case, the encryption/decryption processes may be co-located with the systems and services.

The descriptions in this specification are for purposes of illustration only and are not to be construed in a limiting sense. Therefore, the scope of the invention is limited only by the language of the following claims.